Software Development Methods

# **Objectives**

Introducing concepts and techniques for collaborative development of large and complex software systems for industrial applications, including Java, functional programming, software development lifecycle, best practices in software development as code testing, versioning and design patterns.

### **Knowledge and understanding**

Students should be able to understand the relationships between different software development approaches, software development practices and the cost of changing software. They should have a clear idea about how good practices help in developing software systems and improving their quality.

### **Applied knowledge and understanding**

Students should learn how to use and adopt good software development practices. They should be able to perform test-driven development, refactoring and continuous integration. The students also should learn functional programming techniques and be able to write Java programs that solve practical, real world problems using object-oriented design techniques and data parallel functional programming.

### **Making judgments**

Students should be able to recognize bad code, to identify possible refactoring moves, to analyze a test suite and a deployment pipeline by highlighting qualities and issues.

### **Communication skills**

Students should be able to effectively communicate their design decisions on production code and test code.

### **Learning skills**

Students must demonstrate that they are comfortable with good software development practices. They should demonstrate to understand the basis of Agile Software Development, continuous integration and continuous delivery. They should demonstrate a good understanding on functional programming, writing parallel functional programs in Java and be able to analyze sequential and parallel time complexity in small programs.

# **Prerequisites**

Previous experience with software development and being comfortable with the use of an IDE (or a text editor) and version control systems (like git) are preferable but not mandatory.

# **Contents**

Part 1

* Functional programming: The elements of a functional system: objects, functions and functional forms. Programming examples. Expressive power. Parallel execution considerations.
* Fundamental Java: variables, expressions, control structures, arrays, objects, classes, instance variables, methods, initialization, constructors, inheritance, access control, abstraction, polymorphism, interfaces, exceptions, threads, packages, collections, generics, modules.
* Functional programming in Java: the main elements introduced by the language: method references, lambdas, default methods, streams and optionals. Behavior parametrization. Functional interfaces. Lazy evaluation considerations. Composition. Mutables.
* Java Streams: Main features. Creation, filtering, extraction, grouping, mapping, partition and reducing patterns. Streams vs. Collections. Numeric streams specializations. Performance considerations. Default and custom parallel streams. Split iterators. Comparison of parallel streams with the fork/join framework.

Part 2

* Agile Software Development and cost of change
* Clean code
* Test Driven Development
* Object-oriented design principles
* Testable architectures
* Test automation
* Development collaboration techniques
* Continuous Integration
* Continuous Delivery
* Scrum

# **Teaching methods**

Lectures are expected to be very interactive, with a mix of frontal lectures and hands-on sessions to immediately put into practice the notions, concepts and techniques presented. Hands-on session will include software development exercises and working in pairs on code “katas”. Exercise solutions and “katas” will be discussed after the hands-on sessions. Similar activities may be left has homework and discussed during the next lecture.

# **Evaluation**

The exam will consist in a project to be completed in a group of 2/4 students, using the development practices seen during the lectures.

# **Textbooks**

Part 1

* Java 8 Lambdas, Functional programming for the masses, Richard Warburton, O’Reilly, 2014.
* Beginning Java 9 Fundamentals, Kishory Sharan, Apress, 2017.
* Beginning Java 8 Language features, Kishory Sharan, Apress, 2014.

Part 2

* Clean Code: A Handbook of Agile Software Craftsmanship, Robert C. Martin, Prentice Hall, 2008
* Test-Driven Development: By Example, Kent Beck, Addison-Wesley Professional, 2002
* Growing Object-Oriented Software, Guided by Tests, Steve Freeman and Nat Price, Addison-Wesley Professional, 2009

# **Other information**

Hands-on sessions require a laptop.

Required software:

* Java IDE or text editor, with a preference to the former (we suggest IntelliJ Idea Community)
* Java Development Kit
* Git version control system

Additional suggested books:

* Refactoring: Improving the Design of Existing Code, Martin Fowler, 1999
* Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation, Addison-Wesley Professional, Jez Humble and David Farley, 2010
* Refactoring to Patterns, Joshua Kerievsky, 2004